UDC: 004.2 INFO M: str. 4-14

ARHITEKTURA ORIJENTISANA KA SERVISIMA – WINDOWS COMMUNICATION FOUNDATION (WCF)
SERVICE ORIENTED ARCHITECTURE – WINDOWS COMMUNICATION FOUNDATION (WCF)

Miroslav Minović, Srđan Radojčić

REZIME: U radu je dat pregled osnovnih karakteristika servisno orijentisane arhitekture (SOA), kao arhitekture na kojoj je bazirana i Microsoft-ova nova komunikaciona platforma, Windows Communication Foundation (WCF). WCF predstavlja naprednu infrastrukturu i programski model za izradu distribuiranih aplikacija.
KLJUČNE REČI: SOA, WCF, Komunikacioni paterni, distribuirani sistemi
ABSTRACT: In this paper we present major characteristics of Service-Oriented Architecture (SOA), as a base-ground for Microsoft communication platform, named Windows Communication Foundation (WCF). WCF is an advanced infrastructure and programming model for creating connected applications. It's like nothing that has come before.
KEY WORDS: SOA, WCF, Messaging patterns, Distributed systems

1. UVOD

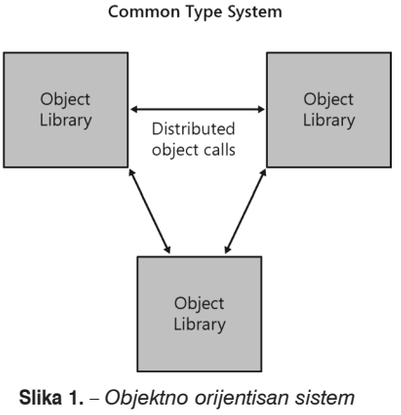
Paradigma klijent – server tehnologije od kada je nastala nije se bitnije promenila, način realizacije te paradigme jeste. U ovom radu biće obrađena nova tehnologija bazirana na Windows Communication Fondation, novi standard prihvaćen od strane 13 vodećih svetskih IT firmi koje definišu razvoj IT tehnologije i standarda danas. Cilj je bio objediniti dosadašnje znanje i postaviti programerski okvir koji će omogućiti implementaciju uobičajenih paterna u klijent server arhitekturi na brz i jednostavan način. Ovim je omogućeno da implementacija same komunikacije oduzme znatno manje resursa, dok se u centar fokusa postavlja aplikaciona logika.

2. ARHITEKTURA ORIJENTISANA KA SERVISIMA (S0)

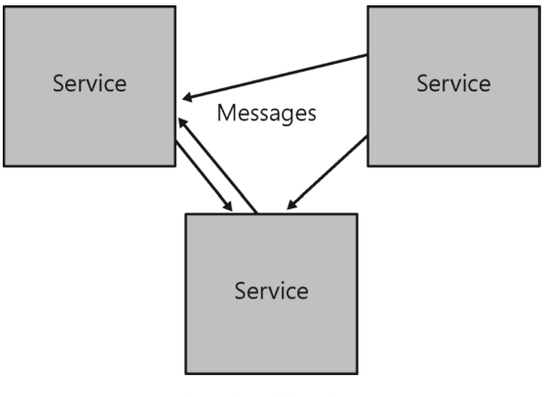
Potreba za povezivanjem ljudi, organizacija, uređaja danas određuje i pravce razvoja arhitekture softvera. Konektivnost je stavljena u centar razvoja softvera. SO je pristup softverskom dizajnu koji je nesumnjivo uzeo primat. Arhitektura orijentisana ka servisima definiše korišćenje servisa kako bi se zadovoljili zahtevi korisnika. Jedna od formalnih definicija opisuje arhitekturu ka servisima kao dizajn za povezivanje resursa (aplikacija ili podataka) na zahtev kako bi se postigli željeni rezultati za korisnika servisa.

WCF (Windows Communication Foundation) predstavlja Microsoftovu SO komunikacionu infrastrukturu i programski model. U SO programi orijentisani ka porukama nazvani servisi predstavljaju gradivne blokove od kojih je rešenje kreirano. WCF predstavlja Microsoftovo izvršno okruženje za servise.

Arhitektura orijentisana ka servisima je komplementarna sa objektnom orijentacijom i ne predstavlja njenu zamenu. Objektna orijentacija ostaje veoma važna u razvoju softvera, ali objekti nisu najbolje rešenje za povezivanje programa u distribuiranom okruženju. Primarna razlika između SO i OO je način na koji je aplikacija sagledana. U zastarelom OO pristupu aplikacija je usko povezana kolekcija programa izgrađenih iz kolekcije (biblioteke) klasa koje imaju međusobne zavisnosti.



Slika 1. – *Objektno orijentisan sistem*



Slika 1. – *SO sistem*

SO aplikacija je sastavljena iz slabo povezanih autonomnih servisa. Delovi sistema komuniciraju preko poruka. Delovi sistema mogu biti u međusobnoj interakciji bez obzira na platformske razlike.

3. OSNOVNI KONCEPTI WCF

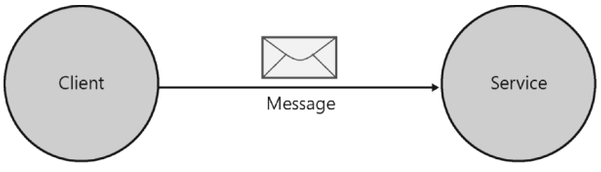
WCF programi komuniciraju kroz razmenu poruka. Postoje 3 različite vrste slanja poruka

- Klijent - program koji inicira poruku
- Servisi - programi koji odgovaraju na poruke
- Posrednici - programi koji su tačke čekanja za poruke ili rute

Iako svaka komunikacija koristi poruke, program ne mora da radi direktno sa porukama. Na primer WCF klijenti mogu pristupiti servisnim operacijama pozivajući metod, prenošenjem parametara za funkciju, i primajući rezultate. Radi se direktno sa porukama samo ako su one definisane i sa zahtevom da se radi na tom nivou.

Klijenti su programi koji inicijalizuju poruke. Klijenti šalju poruke servisinim proogramima i na taj način pokreću izvršavanje serverskih procesa.

Klijent i servis



Slika 3. – *Klijent/Service komunikacija*

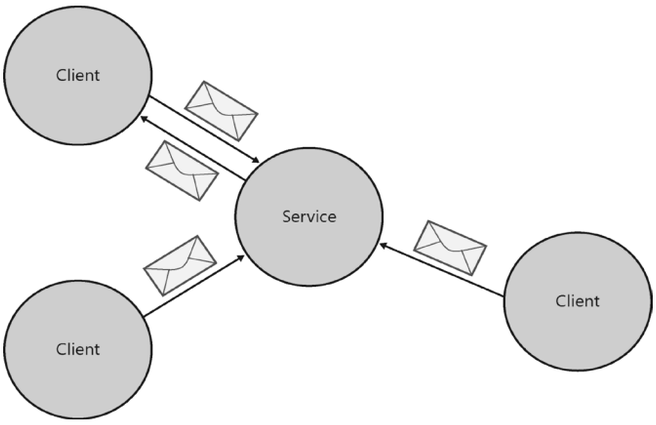
Kada servis primi zahtev od klijenta on čini neke akcije. Klijent takođe može primati poruke. Na primer servis može poslati poruku klijentu kao odgovor na zahtev. Klijent određuje koji će servis kontaktirati i kada će komunicirati sa njim.

Servis

Servisi su programi koji reaguju na poruke. Sve dolazeće poruke pokreću izvršenje nekog ponašanja. To ponašanje može uključiti više poruka kao što je slanje odgovora na izvor dolazeće poruke. Zbog pasivne prirode, servisi ne kontrolišu komunikaciju na način na koji to klijenti čine. Servisi ne mogu odrediti u napred koji će klijent pokušati komunikaciju, kada će se komunikacija desiti ili kakvog će tipa biti komunikacija.

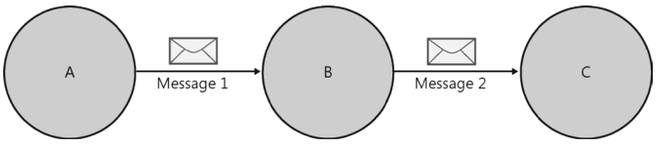
Servisi sa višestrukim klijentima

Tipično servisima pristupa više klijenata istovremeno. Učestalo korišćeni servisi sa mnogo klijenata mogu biti hostovani na servisima. Verovatno je da će servisi imati duže vreme života, dok su klijenti kraćeg veka. Slika prikazuje interakciju servera sa više klijenata, prihvatajući dolazeće poruke i šaljući poruke ka klijentima. Analogno sa web aplikacijama server može održati sesiju za klijenta.



Slika 4. – *Servis sa više klijenata*

Servis takođe može biti i klijent u isto vreme. Dok servis reaguje na dolazeće poruke on može poslati odlazne poruke drugom servisu, u ulozi klijenta. Slika prikazuje lanac komunikacije servisa. Kada program A šalje poruku programu B deo programa B reaguje na ponašanje i šalje poruku programu C. A je klijent dok je C servis. B ima karakteristike i klijenta i servisa.



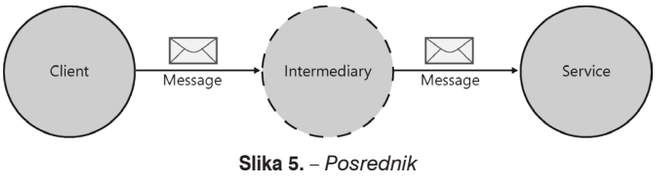
Slika 5. – *Servis kao klijent*

Poruke kroz lanac servisa mogu imati kaskadni efekat.

Posrednici

Komunikacija između klijenta i servisa može prolaziti kroz drugi program nazvan posrednikom. Posrednik je nevidljiv za klijenta i servis i nalazi se između njih. Posrednik može

- o Ponašati se kao firewall, blokirajući neželjene poruke da dođu do servisa
- o Rutiranje poruka na jedan od nekoliko distribucionih servisa
- o Ulogu gejtveja između 2 mreže
- o Praćenje aktivnosti



Slika 5. – *Posrednik*

Posrednik mora da poštuje neka pravila. Posrednici nisu konzumenti poruka, posrednici su samo tačke čekanja za poruke odnosno ruteri do njihovog odredišta. Takođe moguće je da postoje neke informacije u porukama koje posrednik traži ili modifikuje, ali telo poruke, tovar, je rezervisan za krajnjeg korisnika. Razmotrimo

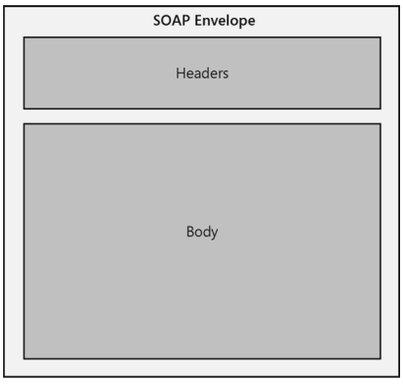
sigurne poruke sa enkriptovanim sadržajem. Poruka ostaje enkriptovana kako prolazi kroz posrednike i samo završni korisnik može pristupiti i dekriptovati sadržaj.

Poruka

Indigo programi svih vrsta klijenta, servisa ili posrednika, komuniciranju kroz razmenu poruka.

Struktura poruke

Spoljašnji koverat sadrži dva dela, zaglavlje i telo. Telo poruke je sadržaj poruke i obično kreiran i konzumiran od klijenta ili servisa aplikacije. Zaglavlje je kolekcija informativnih delova, svaki predstavlja zaglavlje koje može biti korišćeno u aplikaciji, kao i infrastrukturi slanja poruke. Poruka uvek ima jedno telo ali može imati više zaglavlja.



Slika 7. – *SOAP Envelope*

Sve poruke u Indigu su predstavljene kao XML, SOAP koverat sadrži XML infoset. SOAP ne postavlja ograničenje na jedan format ili jedan način slanja poruke. Indigo format i transport je neutralan, sposoban za formatiranje SOAP poruka u nekoliko formata enkodinga (tekst, binarnog, ili MTOM - *Message Transmission Optimization Mechanism*) te slanje preko različitih transportnih mehanizama HTTP, TCP, MSMQ i imenovanih cevi (*named pipes*).

Paterni slanja poruka

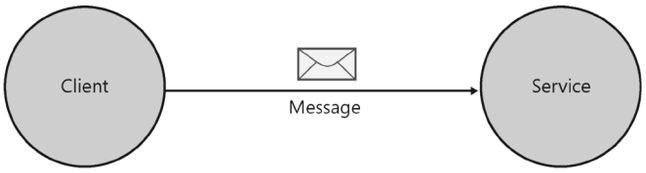
Programi mnogu razmenjivati poruke kroz 3 osnovna paterna:

- Simplex – poruke u jednom smeru
- Duplex – asinhrone poruke u 2 smera
- Request – replay – sinhronizovane poruke u 2 smera

Na osnovu navedenih paterna moguće je konstruisati različite nove načine slanja poruka.

Simplex

Najprostiji patern za razmenu poruka je simplex: poruke u jednom smeru nazvane datagrami koji se razmenjuju među programima. Slanje datagrama od klijenta ka servisu.

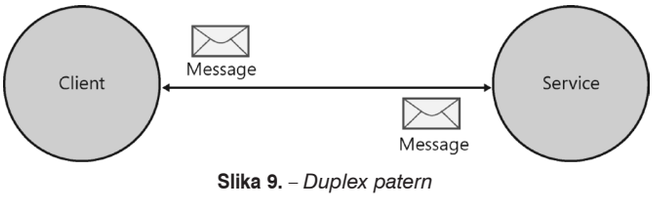


Slika 8. – *Simplex patern*

Simplex komunikacija dozvoljava ispali i zaboravi stil poruka. Ovo je najviša asinhrona forma poruke. Pošalji i zaboravi komunikacija dozvoljava write-only servise, koji su poput crne rupe. Oni prihvataju poruke koje dolaze ali ne šalju poruke.

Duplex

Duplex poruke su poruke u dvosmernoj komunikaciji, u kojoj obe strane mogu komunicirati slobodno, u oba smera, bez sinhronizacije bilo koje vrste.

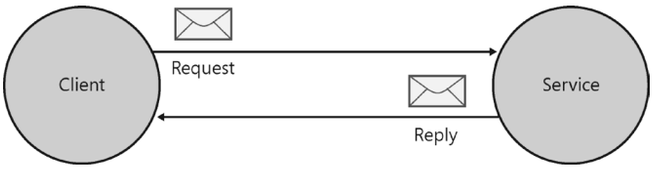


Slika 9. – *Duplex patern*

Primer duplex komunikacije je daljinski kontrolisano vozilo. Komande su poslane ka vozilu kako bi učinilo nešto, vozilo odgovara kako bi odgovorilo na promenu pozicije.

Request-Reply (Zahtev-Odgovor)

Dvosmerna komunikacija u kojoj je zahtev u paru sa odgovorom i čini request-reply patern. Klijent šalje zahtev i čeka za replay.



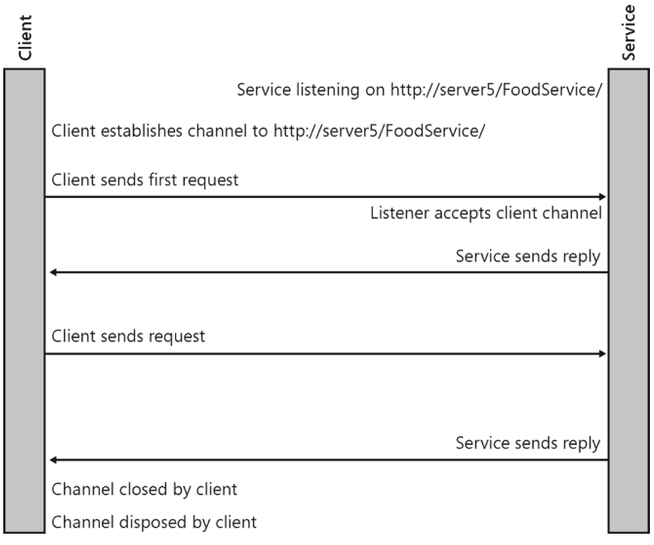
Slika 10. – *Request/Reply patern*

Gotovo svi su upoznati sa ovim paternom, zahvaljući webu. Mnoge tradicionalne komunikacione tehnologije su zasnovane na request-replay paternu, uključujući proceduralne pozive.

Transportni kanali

Kanali služe za prenos poruka. Pre nego što 2 programa mogu razmeniti poruke, kanal mora biti uspostavljen između njih. Osnovni tip WCF kanala implementira simplex ulaz, simplex izlaz, duplex i request-replay patern. Klijent kreira kanal ka krajnjoj tački servisa, definišući tip kanala i adresu. Ako servis sluša na istoj adresi kanal može biti uspotavljen, te

može doći do razmene poruka. Jednom kada je komunikacija završena kanal može biti zatvoren. Kanal nije uspostavljen dok prva poruka nije poslata. Prva poruka od klijenta ka serveru je specijalna na više načina, kanal se uspostavlja u tom trenutku, i bilo koji dogovor oko sigurnosnih protokola se odvija u tom trenutku. Zbog toga prvoj poruci od klijenta ka serveru treba nešto više vremena.



Slika 11. – *Uspostavljanje transportnog kanala*

Zbog toga što kanal nije uspostavljen do slanja prve poruke, ukoliko je bitno testiranje komunikacije pre slanja prve poruke, moguće je izgraditi ping operaciju nad servisom tako da klijent ima način da uspostavi komunikaciju, verifikuje dostupnost servisa i uspostavi sigurnosne parametre ranije.

Stek transportnog kanala

Kanal je kao cev, poruka poslata na jednom kraju cevi pojavljuje se na drugom. Različite cevi mogu biti povezane kako bi formirale stek kanala. Kada WCF program definiše kombinacije potrebnih opcija, traženi stek kanala je kreiran kombinujući odgovarajući miks kanala.

Primer pokazuje request-reply kanal, koji je kombinovan sa enkoderom binarnih poruka, TCP transportni kanal, pouzdani sesijski kanal i windows-ov sigurnosni kanal. Rezultujući kanalni stek obezbeđuje kompozitno ponašanje ali može biti tretiran kao atomska jedinica.

Treba imati u vidu da nije postoje ograničenja prilikom kombinovanja kanala, u zavisnosti od željenog paterna razmene poruka. Na primer duplex komunikacija nije moguća bez pouzdanog kanala sesije.

Transport

Transporti su metodi komunikacija koji kanal koristi, Četiri transportna mehanizma ugrađena u WCF su:

- HTTP,
- TCP,
- MSMQ i
- imenovane cevi (*named pipes*)

Moguće je proširiti listu kreirajući sosptveni transportni provajder. Neki transporti imaju sigurnosne opcije, kao što je HTTPS. Imenovane cevi su namenjene za komunikaciju između procesa na istom računaru, i ne mogu biti korišćene za komunikaciju između računara.

Enkoding

Poruka može biti enkodovana u :

- tekstualni,
- binarni ili
- MTOM format.

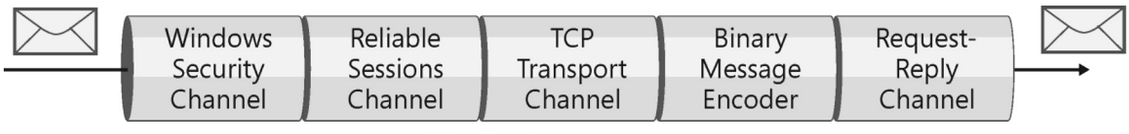
Moguće je proširiti listu implementirajući sopstveni enkoder. Tekst encoding je najmanje efikasan, ali je široko interoperabilan. Binarni ekoding je mnogo efikasniji međutim zahteva .NET platformu sa obe strane konverzacije MTOM koristi specifičan mehanizam da obzebedi visoko optimizovane poruke koje su interportabilne sa drugim platformama koje takođe podržavaju MTOM.

Sigurnost

Sigurnosne opcije mogu uključiti sigurnost u transportu kao što je obezbeđivanje HTTPS i TCPS. Takođe sigurnost postoji na nivou SOAP poruke. Sigurnost poruka može obezbediti autorizaciju, autentifikaciju, auditing, poverljivost poruke, integritet, i detektovanje odgovora. Nekoliko sigurnosnih mehanizama je dosputpno uključujući X.509, Windows NTLM/Kerberos, korisničko ime i šifra. Sigurnost poruka bazirana je na sigurnosnim tokenima. Takođe je moguće dodati novi sigurnosni mehanizam.

Pouzdane sesije

Pouzdani kanal sesija obezbeđuje dve važne mogućnosti, pouzdano slanje poruka i sesije. Pouzdano slanje poruka obezbeđuje komunikaciju u slučaju neuspeha, osiguravajući dostavljanje poruka, kako što je arhiviranje poruka u redu. Sesije dozvoljavaju servisu da održi stanje za klijenta tokom serije interakcija. Ove mogućnosti su obezbeđene na nivou SOAP poruke,



Slika 12. – *Primer steka kanala*

koje dozvoljavaju njihovo korišćenje čak i ako upotrebljeni protokol na transportnom sloju ne podržava pouzdanost ili sesije.

4. INTEROPERABILNOST

U zavisnosti od načina vezivanja zavisi i interoperabilnost sa drugim platformama. Dok korišćenje .NET binarnog formata čini komunikaciju veoma efikasnom, ovaj izbor ograničava komunikaciju na .NET platformu. Postoji nekoliko nivoa interoperabilnosti koje ćemo razmotriti:

• Interoperabilnost osnovnog profila

Predstavlja model interoperabilnost i za servise. Usaglašavanje sa WS-I Basic Profile dozvoljava WCF programima da komuniciraju sa prvom-generacijom Web servisa. Široka interoperabilnost dolazi po cenu da nije moguće predpostaviti podršku druge strane za pouzdanim porukama, sigurnošću ili transakcijama.

• WS-* interoperabilnost

Interoperabilnost sa drugim platformama koje koriste sledeću generaciju WS-* protokola. Osigurano je pouzdano slanje i primanje poruke, sigurnost poruke, i transaktivnost.

• .NET interoperabilnost

Prednost korišćenja .NET platforme na oba kraja komunikacije. Omogućava odlične performance, binarno enkodirane poruke mogu biti korišćene, osigurana je pouzdanost, transaktivnost i sigurnost.

• MSMQ interoperabilnost

Dozvoljava komunikaciju sa MSMQ (*Microsoft Message Queue*) koji nije vezan za WCF.

Standardni način vezivanja obezbeđuje različite nivoe interoperabilnosti. BasicProfile vezivanja dozvoljava platformsku nezavisnost komunikacije sa prvom generacijom web servisa kao što su .NET ASMX web servisi kao i sa sledećom generacijom servisa koja podržava WS-* protokole. Transportna sigurnost je dostupna.

WS-* interportabilnost je obezbeđena WSProfilom i WSProfileDualHttp profilom vezivanja. Omogućava se platformski-nezavisno komuniciranje sa sledećom generacijom servisa koju podržavaju WS-* protokol. Nivo inteportabilnost obezbeđuje platfomrsku nezavisnosti i podržava pouzdane sesije, transakcije, transportnu sigurnost i SOAP sigurnost.

Dot NET interoperabilnost je obezbeđena korz NetProfileNamedPipe, NetProfileMsmq, NetProfileTcp i NetProfileDualTcp profile vezivanja. Dozvoljava .NET ka .NET komunikaciju i podržava transakcije, pouzdane sesije te SOAP sigurnost, interportabilnost je veoma efikasna ali zahteva Microsoft .NET platformu.

Kako je celokupna komunikacija porukama zasnovana na SOAP protokolu, sistemi u komunikaciji moraju se sporazumeti oko:

- Nivoa interoperabilnosti
- Stila dokumenta
- Formata poruke
- Strukture poruke

Stil dokumenta

Opis poruke može biti reprezentovan na nekoliko načina u WSDL-u (*Web Service Description Language*). Može biti u formi dokumenta ili RPC formata. Dokument formata ima 2 varijante :

a) Dokument/wraped u kojoj je sadržaj prikazan kao jedan element

b) document/bare u kojoj nema spoljnih elemenata.

- DocumentWrapped – WDSL koristi format dokumenta, poruke su obmotane.
- DocumentBare – WSDL koristi format dokumenta, poruka nije obmotana
- RPC – WDSL koristi RPC format i poruka sadrzi WS-I Basic Profile i RPC pravila

Kako bi podesili stil dokumenta da bude kompatibilan sa očekivanim sistemom, moguće je definisati stilove parametara u ugovorima servisa [ServiceContract] ili [OperationContract] atributa (Primer 1).

Format poruke

Poruka može biti u potpunosti literalno enkodirana. U tom formatu telo poruke odgovara XML šemi. Endkodirani format koristi SOAP enkoding. Ukoliko je enkodirani format korišćen XmlSerializer mora takođe biti definisan.

Primer 1.

Poruka u Document/Wrapped stilu

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <Add xmlns="http://ProgrammingIndigo">
   <n1>100</n1>
   <n2>15.99</n2>
  </Add>
 </s:Body>
<s:Envelope>
```

Poruka *u Document/Bare stilu*

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <n1 xmlns="http://ProgrammingIndigo">100</n1>
  <n2 xmlns="http://ProgrammingIndigo">15.99</n2>
 </s:Body>
</s:Envelope>
```

Poruka *u RPC stilu*

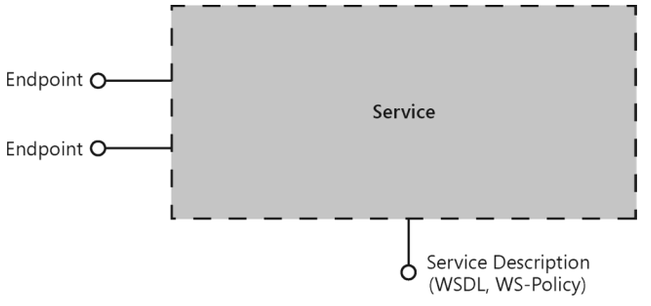
```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <Add xmlns="http://ProgrammingIndigo">
   <n1 xmlns="">100</n1>
   <n2 xmlns="">15.99</n2>
  </Add>
 </s:Body>
```

Struktura poruke

Kako postoji više od jednog načina za čuvanje informacije u SOAP poruci, možda postoji potreba da se izmeni podrazumevani format WCF strukture poruke, kako bi se postigla interportabilnost sa sistemom kojem se pristupa te sa različitom očekivanom strukturom poruke. Ugovor poruke dozvoljava kontrolu nad delovima poruke koji se nalazi u hederu ili telu poruke.

5. ANATOMIJA SERVISA

Servis eksponira opis servisa i kolekciju krajnjih tačaka. Opis servisa govori šta servis može da uradi i kako mu može biti pristupano. Krajnje tačke su pristupne tačke za servise.



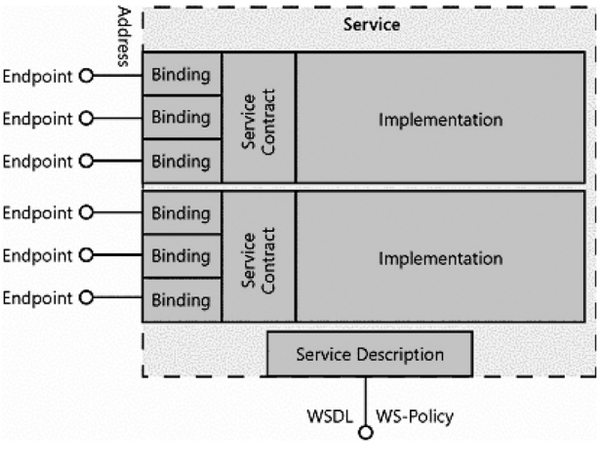
Slika 13. – *Anatomija servisa*

Opis servisa

Opis servisa obezbeđuje osnovne informacije kako servis može biti korišćen. Opisi servisa su izloženi korišćenjem nekoliko standarda koji omogućavaju kompatibilnost servisa sa prvom generacijom web servisa dozvoljavajući sofisticiraniji opis za moderne servise.

- Web Services Description Language (WSDL)
- XML Schema (XSD)
- WS-Policy
- WS-Metadata Exchange (or WS-MEX)

Servis sadrži krajnje tačke, veze, ugovore i implementaciju. Ugovori definišu ponašanje servisa, strukturu, ili format poruka. Veze definišu kako se pristupa servisu. Krajnje tačke povezuju ugovore i veze sa adresama.



Slika 14. – *Struktura servisa: Krajnje tačke, veze, ugovori i implementacija*

Adrese

Servisne krajnje tačke su izložene kroz adrese. Na format adrese utiče više faktora, transport koji se koristi u upotrebi, da li se IIS (*Internet Information Server*) koristi za hostovanje servisa ili se servis hostuje u samostalnoj aplikaciji.

Primer opisa servisa koji daje podatke o akcijama.

```
<?xml version="1.0"?>
<definitions name="StockQuote" targetNamespace="http://example.com/stockquote.wsdl"
      xmlns:tns="http://example.com/stockquote.wsdl"
      xmlns:xsd1="http://example.com/stockquote.xsd"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
        xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float"/>
          </all>
```

```
      </element>
    </schema>
  </types>
  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
  </message>
  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
  </message>
  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
  </portType>
  <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
      <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteBinding">
      <soap:address location="http://example.com/stockquote"/>
    </port>
  </service>
```

Transport	prefiks	Format	Primer
HTTP	http	server[:port]/path/	http://localhost:8000/Service/
TCP	net.tcp	server[:port]/path	net.tcp:// localhost:9000/ Service /
Named pipes	net.pipe	server/path/	net.pipe://localhost/ Service /
MSMQ	net.msmq	server/path/	net.msmq://localhost/private$/ Service/

Ukoliko je servis hostovan pod IIS-om adresa sledi drugačiji format. Servisi hostovani u IIS koriste virtualne direktorijume, koji imaju adresu u formi prefix:// server/virtual-directory/svc-file (npr. http://localhost / Service/report.svc) .Svc fajl sadrži informacije o servisu uključujući i izvorni kod ili reference na kompajlirani kod.

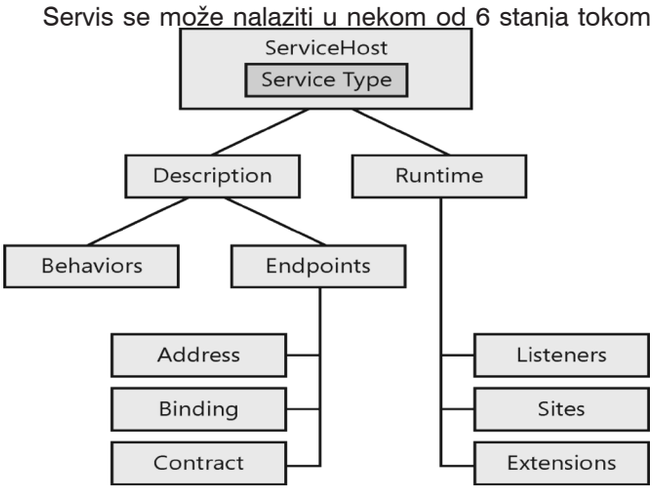
Mnogi koncepti servisa direktno su prezentovani kroz WSDL, ali sa različitim terminima

Service Concept	WSDL Element
Service contract	<portType>
Message contract	<message>
Data contract	<type>
Operation	<operation>
Binding	<binding>
Endpoint	<port>

SERVISI U VREME IZVRŠENJA

Kada se servis u WCF izvršava, servisov opisni i implementacioni kod se prevodi u CLR (*Common Language Runtime*). Kada je poruka primljena ona je dostavljena instanci implementacije servisa. Glavni runtime objekat za servise je nazvan ServiceHost. Preko ServiceHosta pristupa se opisu servisa, ponašanju, krajnjim tačkama, runtime instancama. Neke aspekte servisa moguće je menjati dinamički tokom izvršavanja, kao što je dodavanje novih krajnjih tačaka.

Servis se može nalaziti u nekom od 6 stanja tokom



Slika 15. – *Konceptualni model servisa*

izvršavanja: stanju kreiranja, stanju otvaranja, otvorenog stanja, stanju zatvaranja, zatvorenom stanju, stanju kvara.

Servis može izvršavati neko od sledećih vrsta ponašanja

• Instanciranje

Instanciranje određuje kako su instance implementacije servisa kreirane. Četiri modela instaciranja su: *singleton* – jedna instanca za sve klijente, *po pozivu*- jedna instanca po svakom pozivu operacije*, privatne sesije* - jedna instanca po sesiji klijenta, i *deljene sesije*, jedna instanca po sesiji ali klijenti mogu deliti istu sesiju.

• Konkurentnost

Konkurentnost kontroliše kako su niti i instance povezani. Moguće je implementirarati servisne klase tako da budu bezbedne za niti (*thread safe*) ili mogu zahtevati samo jednu nit u vreme pristupa instance. Tri konkurentna moda su: jedinstvena nit, višestruka nit – više niti pristupa instanci u isto vreme, i reentrant -niti mogu međusobno da komuniciraju. Bitno je naglasiti da prilikom korišćenja callback mehanizma potrebno je koristiti reentrant ili višestruke niti mod, kako ne bi došlo do dedlokova (*deadlock*).

• Rukovanje greškama

Kada dođe do greške moguće je obraditi grešku direktno ili ostaviti da bude obrađena od strane izvršnog okruženja (*framework*).

• Metadata

Servisi mogu biti samoopisujući, obezbeđujući metapodatke na zahtev

• Životni vek

Servisi mogu kontrolisati životni vek klijentske sesije, definišući servisne operacije od inicjalizacije sesije do isteka vremena, kao i prekid sesije

• Sigurnost

Servisima može biti definisano sigurnosno ponašanje, koje uključuje poverljivost poruka, integritet poruka, autentifikaciju, autorizaciju, editovanje, i detekciju odgovora. Sigurnost može biti realizovana preko X.509, username/password, Kerberos, SAML, XrML ili sopstvenog sigurnosnog mehanizma.

• Transakcije

Klijent može poslati transakciju servisu tako da servis može učestvovati i u transakciji. Omogućena je kontrola da li će servis prihvatiti transakciju, klijent kontroliše sklop i životni vek transakcije.

Ponašanje instance servisa

Dok se servis izvršava, jedna ili više instanci implementacionih klasa su kreirane u odgovoru na zahtevanu poruku. Implementacija klasa definiše neki model instanciranja, koji određuje kada će klasa biti kreirana i uništena. Postoje 4 modela instanciranja:

- Po pozivu – instanca klase je kreirana za svaki poziv i zatim uništena
- Singleton – jedna isnatnaca klase obrađuje svaki zahtev
- Privatna sesija – instance klase je kreirana za svaku konekciju klijenta
- Deljena sesija instance klase je kreirana za svaku konekciju klijenta ali je moguće da više klijenata deli pristup istoj instanci

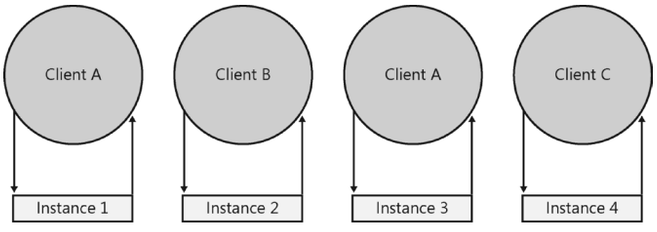
Po pozivu je podrazumevani mod instanciranja, izbor metoda instanciranja zavisi od zahteva za upravljanjem stanjem.

- Instanciranje po pozivu se koristi kada operacije servisa ne zahtevaju čuvanje stanja među pozivima. Na primer per call instanciranje bi bio dobar izbor za servise koji rade nezavisne kalkulacije.
- Singleton instaciranje se koristi kada operacije moraju da očuvaju stanja tokom svih aktivnosti. Primer za upotrebu ovog servisa bi bio servis za glasanje.
- Instancijranje privatne sesije je kada servis treba da očuva stanja nezavisno za svakog klijenta. Primer su ecommerce servisi.
- Instanciranje deljenih sesija je pogodno kada operacije trebaju da održe nezavisna stanja za grupe klijenata. Odličan primer bi bile višekorisničke igre, gde svaka grupa igrača deli instancu koja predstavlja njihovo stanje.

Kada su instancirane privatne sesije i deljene sesije definicije ugovora servisa mogu definisati koje su operacije inicirane, a koje su prekinute. Iniciranje operacija servisa može izazvati nove instance koje trebaju biti kreirane. Termininiranje operacije servisa oslobađa neku instance.

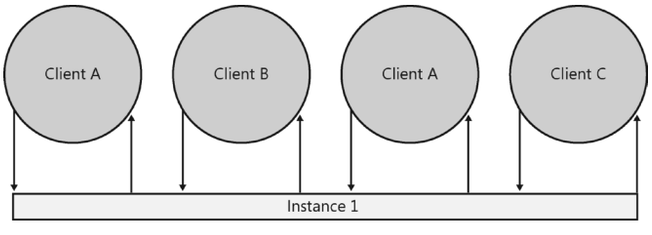
Primer instanciranja

1. Klijent A pristupa servisu
2. Klijent B pristupa servisu
3. Klijent A pristupa servisu drugi put
4. Klijent C pristupa servisu



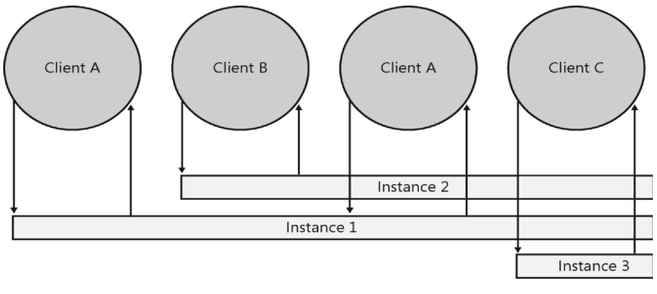
Slika 16. – *Instanciranje po pozivu*

Slika 16 prikazuje instanciranje po pozivu. Svaki klijent pristupa novoj instanci servisovih klasa. Drugi klijent od klijenta A ne koristi ponovo klasu od prvog poziva. Svaka instanca je oslobođena po završetku operacija servisa.



Slika 17. – *Singleton instanciranje*

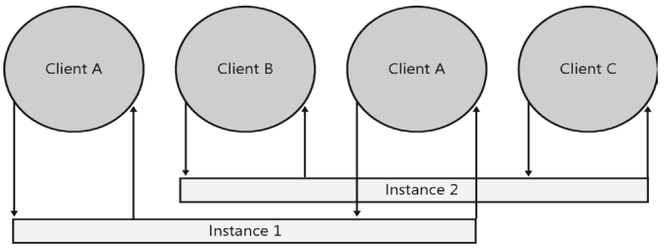
Svi klijenti koriste istu instancu servisa. Prilikom startovanja servisa instanca je kreirana.



Slika 18. – *Instanciranje privatnih sesija*

Stanje klijenta je očuvano među pozivima, tokom iste konekcije klijenta. Svi pozivi klijenta A koriste istu instancu od prvog poziva. Instanca za svakog klijenta se ubija kada je veza zatvorena, ili kada klijent prekine operaciju servisa.

Klijent B i klijent C dele istu instance sesije. Klijent B održava jedinstvenu instancu i deli instancu sa klijentom C. Dok korišćenjem instanciranja privatnim sesija stanje



Slika 19. – *Instanciranje deljenih sesija*

biva očuvano među pozivima klijenta. Drugi poziv od klijenta koristi istu instancu kao za prvi poziv. Klijent B i klijent C dele istu sesiju.

Servis može definisati kako će svaka instanca biti obrađena u odnosu na niti. Implemtacija klase određuje konkurentni mod, koji definiše koliko će niti biti dozvoljeno za simultani pristup servisima instance, Postoje 3 konkurentna moda

- Single – jedna nit u vreme kada je servis instanciran
- Reentrant – jedna nit u vreme pristupa instanci servisa i call-back instanci servisa su dozovljeni.
- Multiple – više od jednog konkurentnog pristupa instanci servisa

Throttling

Throttling ponašanje dozvoljava kontrolu preko količine zahteva koju servis dobija. Ovo omogućava da se setuje ograničenje kako bi se izbeglo preopterećenje resursa servisa Throttling obezbeđuje kontrolu nad 4 dela

- Broj poziva ka servisu
- Broj konekcija ka servisu
- Broj instanci servisa
- Broj zahteva na čekaju na servis

Master klasa koja reprezentuju servise u vreme izvršavanja je ServiceHost<T>. Objekat ServiceHost sadrži tipove servisa. Kada je taj objekat kreiran, tip servisa je definisan kao generički.

ServiceDescription

ServiceHost sadrži opis servisa koji koristio ServiceDescritpiton klasu. **ServiceDescritpiton** sadrži informacije o servisu koje mogu biti izložene u obliku metapodatka. Ovo uključuje njegove ugovore, šeme i definicije krajnjih tačaka.

ServiceSite

Prilikom izvršenja servisa, kreiraju se višestruke instance tipova servisa u zavisnosti od aktivnosti klijenta. ServiceSite povezuje kanal sa instancom tipa servisa. ServiceSite omogućuje da kanal i instance servisa ima različita vremena života. ServiceHost održava kolekciju servisa koristeći ServiceSite klasu.

OperationContext

Operacioni kontekst je očuvan za operacije servisa koje mogu biti korišćene da lociraju servisov host, servisov sajt, dolazeće informacije i sigurnosni sadržaj.

ServiceSecurityContext

Takođe možda će biti potrebno da servis sazna nešto o sigurnosnom okruženju pod kojim je startovan. Sigurnosni kontekst je očuvan za operacije servisa. Servisove operacije mogu koristiti sigurnosni kontekst da odrede informacije kao što je identitet onoga ko pristupa servisu. Sigurnosnom kontekstu je pristupljeno kroz ServiceSecurityContext klase.

Poruke i operacije servisa

Kada je poslata validna poruka za servis, ona je prosleđena metodu koji implementira operacije servisa. Metod može pristupiti podacima iz poruke sa zahtevom kroz ulazne parametre. Ako je operacija servisa dvosmerna, postojaće poruka sa odgovorom. Metod može definisati podatke za odgovor kroz rezultate metoda ili kroz izlazne parametre.

6. ZAKLJUČAK

Web servisi kao relativno nov način izvršavanja poziva udaljenih metoda preko HTTP protokola pri čemu je moguće koristiti SOAP rešili su probleme kojii su se javljali korišćenjem DCOM platforme. Interportabilnost omogućuje potpunu nezavisnost klijenta od implementacije servisa. Windows Comunication foundation predstavlja korak dalje, omogućujući pouzdanost, sigurnost, i interportabilnost. WCF će zasigurno postaviti nove standarde u projektovanju i implementaciji distribuiranih aplikacija.

Windows communication fondation sa druge strane predstavlja subliminat znanja i iskustva stečenog u razvoju distribuiranih aplikacija, te nastojanje da se uobičajeni paterni koji se javljaju u svakoj distribuiranoj aplikaciji realizuju na što jednostavniji način, ostavljajući tako više vremena za suštinu problema koja aplikacija rešava.

LITERATURA

[1]. Minović, M., Radojčić,S.: "INDIGO – servisno orijentisana komunikacija", zbornik radova, InfoTech 2006, Vrnjačka Banja

[2]. McMurtry,C., Mercuri, M., Watling, N: "Windows Communication Foundation: Hands-On

[3]. Pallman, D. *Programming INDIGO*, Microsoft Press, 2005

[4]. Herzum, P., *Web Services and Service-Oriented Architectures*. Executive Report, vol. 4, no.10.

[5]. Cutter Distributed Enterprise Architecture Advisory Service, 2002

[6]. Bieber, G., and Carpenter, J. *Introduction to Service-Oriented Programming* (Rev 2.1). www.openwings.org/download/specs/ServiceOrientedIntroduction.pdf

[7]. http://msdn.microsoft.com/winfx/technologies/communication/default.aspx (Indigo home page)

[8]. Windows Communication Foundation http://wcf.netfx3.com

Miroslav Minović, dipl. ing. asistent-pripravnik Fakulteta organizacionih nauka u Beogradu.
Oblasti interesovanja: mobilno računarstvo, distribuirani sistemi, multimedijalni sistemi i komunikacije.

Srđan Radojčić, dipl. ing. Saradnik laboratorije za Multimedijalne komunikacije Fakulteta organizacionih nauka u Beogradu.
Oblasti interesovanja: softversko inženjerstvo, informacioni sistemi, multimedijalni sistemi.